REMARKS

In view of the following remarks, Applicant respectfully requests reconsideration and allowance of the subject application. This amendment is believed to be fully responsive to all issues raised in the October 27, 2006 Office Action.

Rejections to the Claims

35 U.S.C. 112

Claims 1-23 and 53-63 remain rejected under 35 U.S.C. 112, first paragraph, as filing to comply with the enablement requirement. In response to the Applicant's arguments in reference to whether or not the term "focusable" is will-known in the art, the Examiner maintains that the term is not properly defined by the specification, and evidence of the existence of the term in a document published nearly two years later does not prove it to be well-known in the art. Applicant respectfully traverses this rejection.

In the field of user interfaces, to which the subject application applies, the terms "focus" and "focusable" are well-known in the art, and have been since a time prior to the filing date of the subject application. One example reference that discusses focusable elements within a user interface is VISUAL BASIC 6 WEEKEND CRASH COURSE, by Richard Mansfield, copyright 2000 (herein referred to as "Mansfield"). On page 33, in the section titled, "Courtesies for the Keyboard-bound: Adjusting the TabIndex Property," Mansfield states:

Some people like to use the Tab key to move between the components. Each time they press Tab, the focus moves to the next component (as defined by the TabIndex property of each component). Pressing the spacebar triggers the component that currently has the focus. For example, when one of your CommandButtons has the focus, pressing the spacebar will depress that button, just as if the button had been clicked with the mouse.

On page 33, Mansfield also states, "You can only type into a TextBox if it has the focus."

On pages 77-78, Mansfield states:

The TabStop property, when set to False, removes the component from the TabIndex list. TabIndex is useful because it offers a quick way for the user to move among the input components (TextBoxes, CheckBoxes, and so forth) on a form – all without having to remove his or her hands from the keyboard and reach for the mouse to click a component into the focus. However, there are some components, like a PictureBox, that are not usually employed as user-input devices. So you can set their TabStop properties to False to eliminate them from the TabIndex group. Components such as Labels that can never be used as input devices simply have no TabIndex property in the first place, and are therefore never included in the tabbing.

How can a PictureBox ever be used as an input device, you ask? A simple example is that if the PictureBox is clicked at all, anywhere, something happens (because you put some programming into its Click event). You might display several small PictureBoxes, each containing a different image – perhaps a car,

a bus, a train, or a plane. When you click one, a phone number where you can arrange for that kind of transportation is displayed.

Finally, on page 273, Mansifeld states, "They also expect to be able to press the Tab key to move among the components. The order of the TabIndex properties determines the order in which each component gets the focus."

As discussed above, Mansfield clearly shows that the terms "focus" and "focusable" are well-known to those skilled in the art. Furthermore, the application defines the term "focusable" by way of example, and the examples given in the application are consistent with the use of the terms "focus" and "focusable" as discussed in Mansfield. Specifically, paragraph [0031] of the application states:

> When accessibility mode is activated, elements that are not, by default, focusable, but that have associated alternate content, are added to a list of focusable elements that is maintained by rendering engine 214 based on the specification associated with the document. In the illustrated example, the title text element 302 and image element 310 are not focusable elements by default, but have associated alternate content, and are thus added to a list of focusable elements that includes button element 308.
>
> ...
>
> If a viewer selects the text element that currently has focus (e.g., by pressing a "select" button on the remote control), the alternate content associated with the element is rendered.

Accordingly, as shown above, the term "focusable" is, in fact, a term that is well-known to those skilled in the art. As such, Applicant respectfully requests that the 112 rejection of claims 1-23 and 53-63 be withdrawn.

35 U.S.C. 102(b)

Claims 1-15, 20, 21, 23, 24, 53-55, 57, 58, 62, and 63 remain rejected under 35 U.S.C. 102(b) as being anticipated by U.S. Patent Application Publication Number 2002/0152283 filed by Dutta et al. (herein referred to as "Dutta"). Applicant respectfully traverses this rejection.

The Application describes providing access to alternate content in documents rendered using an interactive television viewing system. A browser or other rendering engine that supports access to alternate content determines which elements in the document may receive focus, based on whether or not an accessibility mode is active. When an accessibility mode is active, the list of focusable elements may contain elements that, by default, are not focusable elements, but that have associated alternate content. An icon or other demarcation, either audio or visual, can be used to indicate to the user whether or not an accessibility mode is active. Furthermore, an input device, such as a television remote control can be configured to provide a mechanism for activating and deactivating an accessibility mode. (*Application, Summary.*)

The Application further notes that some hypertext markup language (HTML) browser applications written for use on computer systems include functionality for accessing alternate content, but typically rely on pointing

devices, such as a mouse. Unfortunately, interactive television systems generally support user interaction through a television remote control and do not include a pointing device. As such, interactive television systems lack a mechanism for allowing viewers to access alternate content in HTML documents. (*Application*, paragraph [0002].)

As illustrated in Figure 1, the Application is clearly directed to a system that does not include a pointing device, such as a mouse. Paragraph [0017] states, "navigation is performed by a user using directional inputs by pressing arrow buttons 114 of remote control 108, arrow keys or other inputs associated with handheld device 110, or arrow keys 116 of keyboard 112." Paragraph [0017] further states, "pressing a 'select' button on remote control 108 while a button associated with an HTML document has the focus, causes the browser to perform the functionality associated with the button."

Specifically, claim 1 recites:

> A method comprising:
>
> receiving a document to be rendered, the document including a first element that has standard content and is focusable and a second element that has both standard and alternate content and is not focusable;
>
> generating a list of focusable elements associated with the document, such that the list includes the first element, but does not include the second element;
>
> rendering the document such that the standard content associated with the first element is rendered and the standard content associated with the second element is rendered;

receiving an indication of a viewer's intent to activate an accessibility mode; and

altering the list of focusable elements to include the second element.

Dutta does not show or disclose "generating a list of focusable elements associated with the document, such that the list includes the first element, but does not include the second element;" and "altering the list of focusable elements to include the second element," as recited in claim 1.

Dutta discloses providing access to alternate formats within an electronic document. A web page is parsed and a document object model (DOM) is created, and the alternate format attribute of an image element within the DOM is parsed. The browser then displays the web page containing an image (or images) according to the default settings of the browser. A user interface is specified which allows the user to select alternate formats for the default image. This user interface may be in the form of a pop up menu that is presented to the user in response to an input command, such as a right mouse click on the default image. The user can then select an alternate format which replaces the original image in the web page. (*Dutta, Abstract.*) However, Dutta does not disclose, "generating a list of focusable elements," and, "altering the list of focusable elements," as recited in claim 1.

In the Response to Arguments section (*Office Action*, page 11-12), the Office states, "Dutta discloses that the document is rendered with the standard content of both the first and second elements and only the elements which are

interactive are currently focusable." The Office further states, "The browser properly applies interaction capability to only those items which are focusable, thus the browser in affect has created a virtual list." In support of these statements, the Office cites Dutta, paragraphs [0005]-[0008], [0038]-[0049], and [0050-0053]. (*Office Action*, page 11.)

Dutta, paragraphs [0005]-[0008] describe existing systems for accessing alternate content, in which an alternate format can be applied to an entire page to view the alternate content. The cited portion also provides a summary that describes providing access to alternate formats within an electronic document. Specifically, paragraph [0008] states:

> A user interface is specified which allows the user to select alternate formats for the default image. This user interface may be in the form of a pop up menu that is presented to the user in response to an input command, such as a right mouse click on the default image. The user can then select an alternate format which replaces the original image in the web page.

Dutta, paragraphs [0038]-[0049] describe providing an "ACTIVE" ALT tag for a multimedia resource that gives the user the ability to choose an appropriate format from a list of alternate formats. The list can be a pop up menu, an audio list, or a tactile menu. The tag can be activated by right clicking or double clicking on a mouse, or by any other means by which options and menus are usually accessed.

Dutta, paragraphs [0050]-[0053] describe an example in which a user right clicks on a video clip, causing a pop up menu to appear. The pop up

menu contains alternate formats such as audio, jpeg, and text. The user may then select an alternate format from the menu which will replace the default video clip in the display. Paragraph [0052] also indicates that various methods for accessing pop up menus exist and may be used instead of right clicking a mouse.

Dutta may describe a document having a first element that has standard content and is focusable and a second element that has both standard and alternate content and is not focusable. However, Dutta does not describe generating a list of focusable elements, and then altering the list of focusable elements to include the second element. Rather, Dutta describes parsing the document to generate a document object model (DOM), which is a parse tree, in which the various structural elements and relationships among the elements are apparent from the topology of the parse tree. (Dutta, [0035].) As described by Dutta, the DOM includes ***all*** of the elements of the document, and is not, "a list of focusable elements associated with the document, such that the list includes the first element, ***but does not include the second element***," as claimed. It appears that the Office is suggesting that the image described in Dutta is an element that is initially not focusable, but is later added to a list of focusable elements. Applicant respectfully disagrees. Applicant agrees that the image described in Dutta is not a focusable element, but Dutta does not describe modifying a list of focusable elements to include the image. Rather, Dutta describes adding java script in association with the image such that the java script causes a menu to be presented to the user when the user right-clicks

or double-clicks on the image. Executing java script when the image is double-clicked or right-clicked does not imply that the image is a focusable element. In fact, the execution of the java script sets the focus to the menu that is presented to user, enabling the user to select an item from the menu. The image itself is not described by Dutta as ever having the focus or as being a focusable element.

Even according to the Examiner's definition of focusable (i.e., "viewable and selectable" – see Office Action, page 2), Dutta does not indicate that the image is ever selectable – only that javascript is executed if a right-click or double-click is performed while the cursor is located over the image. The image itself is never described as having been selected.

For the reasons stated above, Dutta does not disclose, "receiving a document to be rendered, the document including **a first element that** has standard content and **is focusable** and **a second element that** has both standard and alternate content and **is not focusable**; generating a list of focusable elements associated with the document, **such that the list** includes the first element, but **does not include the second element**; and altering the list of focusable elements to include the second element," as claimed. Accordingly, claim 1 is allowable over Dutta, and Applicant respectfully requests that the 102 rejection be withdrawn.

Claims 2-15 are allowable at least by virtue of their dependence (direct or indirect) on claim 1. One or more of claims 2-15 may also be allowable for independent reasons. For example:

Claim 3 recites, "The method as recited in claim 1 wherein the second element comprises a text element."

With reference to claim 3, the Office cites Dutta, Figure 8 and paragraph [0005]. According to Dutta, paragraph [0017], "FIG. 8 depicts a pictorial diagram illustrating ALT menu on a web page in accordance with the present invention." With reference to FIG. 8, Dutta, paragraph [0049] states that, "the web page 800 has two multimedia resources: a video clip 801 and an image 802." Neither of the elements are described as being a text element. Dutta, paragraph [0005] states, "Web pages are often created with various multi-media resources such as Video Clip, Audio, and Images."

While FIG. 8 may illustrate that the *alternate content* may be formatted as text, the cited portions of Dutta does not illustrate or describe a "second element comprises a text element," as claimed. Accordingly, for this reason, in addition to its dependence on claim 1, claim 3 is also allowable over Dutta.

Claim 11 recites, "The method as recited in claim 1 wherein the alternate content comprises a second document that can be rendered."

With reference to claim 11, the Office cites Dutta, Figure 8 and paragraph [0005]. As described above with reference to claim 3, the cited

portions of Dutta describe various multimedia resources with which web pages are often created. Paragraph [0005] makes no mention of alternate content, and does not describe "alternate content comprises a second document that can be rendered," as claimed. FIG 8 illustrates example pop up menus for selecting a format of alternate content, but neither menu 811 or 812 indicate that, "the alternate content comprises a second document that can be rendered." Accordingly for this reason, in addition to its dependence on claim 1, claim 11 is also allowable over Dutta.

Claim 13 recites, "The method as recited in claim 12 wherein the rendering the alternate content is performed in response to receiving an indication that the second element has been selected."

Claim 14 recites, "The method as recited in claim 13 wherein the indication that the second element has been selected comprises an indication that the element has focus."

With reference to claims 13 and 14, the Office cites Dutta, paragraphs [0008], [0038]-[0049], and [0050-0053]. The Office also states, "Dutta discloses that the alternate content of an element is rendered when selected, and that an indication of selection is shown." (Office Action, page 4.) Applicant respectfully disagrees.

Regarding claim 13 and the Office's statement that, "Dutta discloses that the alternate content of an element is rendered when selected," Applicant would like to point out that the Office has not stated that the alternate content is

rendered when, "the second element has been selected," as claimed, but rather that, "the alternate content of an element is rendered when selected." In fact, Dutta describes rendering the alternate content when a format of the alternate content is selected from a pop up menu. Dutta does not describe rendering alternate content in response to receiving an indication that the ***element*** has been selected. As discussed above with reference to claim 1, Dutta describes adding java script in association with the image such that the java script causes a menu to be presented to the user when the user right-clicks or double-clicks on the image. Executing java script when the image is double-clicked or right-clicked does not imply that the image itself has been "selected", but rather that an OnClick event has been launched.

Regarding claim 14 and the Office's statement that Dutta discloses that an indication of selection is shown, the Office has not stated that Dutta discloses that, "the indication that the second element has been selected comprises an indication that the element has focus," as claimed. As discussed above with reference to the 112 rejections, Dutta does not describe a focusable multi-media element. Furthermore, Dutta does not illustrate or describe indicating that an element has the focus.

Accordingly, for these reasons, and by virtue of their dependence (direct or indirect) on claim 1, claims 13 and 14 are also allowable over Dutta.

With reference to claims 20, 21, and 23, the Office has rejected claims 20, 21, and 23 for the same reasons given for claim 1. However, claims 20, 21,

and 23 depend from claim 1, and are therefore allowable at least by virtue of their dependence on claim 1.

Regarding independent claims 24 and 53, the Office states that Applicant's arguments have been considered but are moot in view of the new ground(s) of rejection. However, Applicant notes that no new grounds of rejection have been presented with reference to claims 24 and 53. Accordingly, for the reasons stated in response to the previous Office Action, claims 24 and 53 remain allowable, and Applicant respectfully requests that the 102 rejections be withdrawn.

Claim 54 is allowable by virtue of its dependence on claim 53. Furthermore, claim 54 is also allowable for independent reasons. For example, claim 54 recites, "a focusable element identifier for determining which of a plurality of elements associated with the document are focusable." With regard to claim 54, the Office cites Dutta, paragraphs [0005]-[0008] and [0038]-[0053]. As discussed above with reference to the 112 rejections, Dutta does not describe, "determining which of a plurality of elements associated with the document are focusable." Dutta describes a document that includes elements that may or may not have alternate content. Dutta makes no distinction between whether or not any particular ones of the elements are focusable. This is understandable because Dutta describes accessing alternate content via a pop up menu that is launched in response to a mouse click over an element. In

contrast, the application describes a system in which a mouse is not present, as such, an alternate method is employed (i.e., modifying a list of focusable elements) to enable a user to access alternate content associated with elements that are, by default, not focusable.

Accordingly, for these reasons, and by virtue of its dependence on claim 53, claim 54 is also allowable over Dutta.

Claim 55 recites, "means for modifying a list of focusable elements associated with a document to be rendered."

As discussed above with reference to claim 1, Dutta does not describe modifying a list of focusable elements associated with a document. Accordingly, claim 55 is allowable over Dutta, and Applicant respectfully requests that the 102 rejection be withdrawn.

Claim 57 recites, "determine a list of focusable elements associated with a document to be rendered;" and, "alter the list of focusable elements to include elements that have associated alternate content."

As described above with reference to claim 1, Dutta does not disclose altering a list of focusable elements associated with a document to be rendered. Accordingly, claim 57 is allowable over Dutta, and Applicant respectfully requests that the 102 rejection be withdrawn.

Claims 58, 62, and 63 are allowable over Dutta at least by virtue of their dependence on claim 57. One or more of claims 58, 52, and 63 may also be allowable for independent reasons.

35 U.S.C. 103(a)

Claims 16 and 59-61 remain rejected under 35 U.S.C. 103(a) as being unpatentable over Dutta in view of Microsoft Computer Dictionary, published in 2002, fifth edition (herein referred to as "MCD"). Applicant respectfully traverses this rejection.

Claims 16 and 59-61 are allowable as depending from a respective allowable base claim and for their own recited features which are neither shown nor described in the references of record. To the extent that claims 1, 12, 57, and 58 are allowable over Dutta, the further rejection of claims 16 and 59-61 over the reference to MCD is not seen to add anything of significance because MCD does not cure the deficiencies of Dutta. Accordingly, Applicant respectfully requests that the 103 rejection be withdrawn.

Claims 17-19 and 56 remain rejected under 35 U.S.C. 103(a) as being unpatentable over Dutta in view of U.S. Patent Number 6,819,961 issued to Jacobs et al. (herein referred to as "Jacobs"). Applicant respectfully traverses this rejection.

Claims 17-19 and 56 are allowable as depending from a respective allowable base claim and for their own recited features which are neither shown nor described in the references of record. To the extent that claims 1 and 55 are allowable over Dutta, the further rejection of claims 17-19 and 56 over the reference to Jacobs is not seen to add anything of significance because Jacobs does not cure the deficiencies of Dutta. Accordingly, Applicant respectfully requests that the 103 rejection be withdrawn.

Claim 22 remains rejected under 35 U.S.C. 103(a) as being unpatentable over Dutta. Applicant respectfully traverses this rejection.

Claim 22 is allowable as depending from an allowable base claim and for its own recited features which are neither shown nor described in the references of record. As discussed above, claim 1 is allowable over Dutta. Accordingly, by virtue of dependence, claim 22 is also allowable over Dutta, and Applicant respectfully requests that the 103 rejection be withdrawn.

Conclusion

Claims 1-24 and 53-63 are believed to be in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the present application. Should any issue remain that prevents immediate issuance of the application, the Examiner is encouraged to contact the undersigned agent to discuss the unresolved issue.

Respectfully Submitted,
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201

Dated: 2/22/07

Kayla D. Brant

Name: Kayla D. Brant
Reg. No. 46,576
Phone No. (509) 324-9256 ext. 242

VISUAL BASIC® 6

WEEKEND CRASH COURSE

RICHARD MANSFIELD

Author of *Visual InterDev™ 6 Bible*

15 HOURS

CD-ROM includes an assessment test and all sample code

Bonus Visual Basic 6 Working Model Edition

30 Step-by-Step Lessons That Will Have You Programming in Only 15 Hours

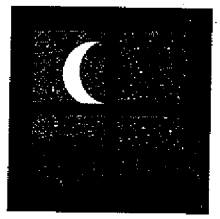
Visual Basic® 6 Weekend Crash Course
Published by
M&T Books
An imprint of IDG Books Worldwide, Inc.
919 E. Hillsdale Blvd., Suite 400
Foster City, CA 94404
www.idgbooks.com (IDG Books Worldwide Web site)

ISBN: 0-7645-4679-1
Printed in the United States of America
10 9 8 7 6 5 4 3 2
1B/QX/QS/QQ/FC
Distributed in the United States by IDG Books Worldwide, Inc.
Distributed by CDG Books Canada Inc. for Canada; by Transworld Publishers Limited in the United Kingdom; by IDG Norge Books for Norway; by IDG Sweden Books for Sweden; by IDG Books Australia Publishing Corporation Pty. Ltd. for Australia and New Zealand; by TransQuest Publishers Pte Ltd. for Singapore, Malaysia, Thailand, Indonesia, and Hong Kong; by Gotop Information Inc. for Taiwan; by ICG Muse, Inc. for Japan; by Intersoft for South Africa; by Eyrolles for France; by International Thomson Publishing for Germany, Austria, and Switzerland; by Distribuidora Cuspide for Argentina; by LR International for Brazil; by Galileo Libros for Chile; by Ediciones ZETA S.C.R. Ltda. for Peru; by WS Computer Publishing Corporation, Inc. for the Philippines; by Contemporanea de Ediciones for Venezuela; by Express Computer Distributors for the Caribbean and West Indies; by Micronesia Media Distributor, Inc. for Micronesia; by Chips Computadoras S.A. de C.V. for Mexico; by Editorial Norma de Panama S.A. for Panama; by American Bookshops for Finland.

For general information on IDG Books Worldwide's books in the U.S., please call our Consumer Customer Service department at 800-762-2974. For reseller information, including discounts and premium sales, please call our Reseller Customer Service department at 800-434-3422.
For information on where to purchase IDG Books Worldwide's books outside the U.S., please contact our International Sales department at 317-596-5530 or fax 317-596-5692.
For consumer information on foreign language translations, please contact our Customer Service department at 800-434-3422, fax 317-596-5692, or e-mail rights@idgbooks.com.
For information on licensing foreign or domestic rights, please phone +1-650-655-3109.
For sales inquiries and special prices for bulk quantities, please contact our Sales department at 650-655-3200 or write to the address above.
For information on using IDG Books Worldwide's books in the classroom or for ordering examination copies, please contact our Educational Sales department at 800-434-2086 or fax 317-596-5499.
For press review copies, author interviews, or other publicity information, please contact our Public Relations department at 650-655-3000 or fax 650-655-3299.
For authorization to photocopy items for corporate, personal, or educational use, please contact Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, or fax 978-750-4470.

Library of Congress Cataloging-in-Publication Data
Mansfield, Richard, 1945–
Visual Basic 6 weekend crash course / by Richard Mansfield.
p. cm.
ISBN 0-7645-4679-1 (alk. paper)
1. Microsoft Visual BASIC. 2. BASIC (Computer program language) I. Title.
QA76.73.B3 M36492 2000
005.26'8--dc21 99-059505

SESSION



Talk to Your User: Creating an Interface

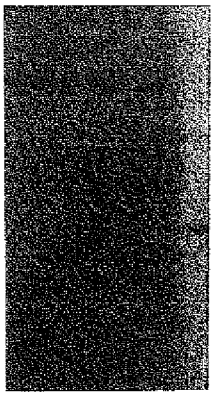
Session Checklist

- ✔ Understanding Rapid Application Development (RAD)
- ✔ Manipulating the properties of a TextBox
- ✔ Listing the features of your application
- ✔ Adding tooltips to assist the user
- ✔ Running and testing the application
- ✔ Saving the application
- ✔ Adjusting the `TabIndex` properties

30 Min. To Go

Nowadays, the term *Visual* is in the names of dozens of languages and products — Visual C++, Visual Studio, and so on. But back in 1991, Visual Basic was the first language to offer what's now called *RAD* features. RAD stands for Rapid Application Development, and it means a collection of tools that considerably lighten a programmer's burden. What's more, working with RAD is often just plain *fun* (don't tell the boss).

VB has had nearly a decade now to perfect its RAD tools. Those tools — along with VB's English-like vocabulary and syntax — are the primary reason that Visual Basic is by far the world's most popular programming language.



When you're finished running the application and playing with the tooltips, shut the application down by clicking the X icon in the upper-right corner of Form1 — or select Run ⇨ End in the VB editor window.

Saving Your Work

You've gone to the effort to make a nice, thoughtful user interface. In the process, as a welcome byproduct, you've also defined what your program will do for the user when it's finished.

Now save your project to your hard drive. If lightning strikes and you lose power between now and Session 4, you won't have to repeat Session 3. When you're ready for Session 4, you can just reload the project back into VB. In Session 4, you will fill in the programming that makes all your CommandButtons actually do what their captions say they can do.

To save your project, first run Windows Explorer and create a new folder on your hard drive named *CCourse1*. Now, in VB, choose File ⇨ Save, browse your hard drive until you find the CCourse1 folder, and then click the Save button on the Save File As dialog box. This saves Form1. Now you must click the Save button a second time to save the project (Project1). Finally, the dialog box closes, and your work is secure.

Done!

Courtesies for the Keyboard-bound: Adjusting the TabIndex Property

You should add two final niceties to your application before considering the user interface finished. Some people like to use the Tab key to move between the components. Each time they press Tab, the focus moves to the next component (as defined by the `TabIndex` property of each component). Pressing the spacebar triggers the component that currently has the focus. For example, when one of your CommandButtons has the focus, pressing the spacebar will depress that button, just as if the button had been clicked with the mouse. Some people prefer to leave their hands on the keyboard when using a word processor, so you always want to take into account the `TabIndex`.

By default, each newly added component gets the next higher `TabIndex`. However, recall that you rearranged the buttons after creating them. So the `TabIndexes` are scrambled. In this application, you want to have the tab simply move down the buttons in order. So leave the TextBox's `TabIndex` property alone — it's 0 (the first, lowest index number), and that's fine. That

The two properties whose names begin with OLE can be safely ignored. They tell VB whether you, the programmer, or the component itself is supposed to deal with dragging and dropping activity. Again, I recommend you avoid the whole drag-and-drop behavior in your VB projects.

The PasswordChar property enables you to specify which character should appear visible to the user when he or she types in a password. In other words, if you want to use a TextBox as a password-entry field for the user, you can type in a * symbol as the PasswordChar. If you type in any character as the PasswordChar, the TextBox will display only that character as the user types: **********, like that. You know the routine. I've always wondered whether this subterfuge is all that necessary. After all, do you have people hovering over your shoulder all the time, just waiting to see your password? I suppose it's better to hide it though. There *are* lurkers.

Note that the MultiLine property must be set to False for the password feature to work properly.

For an English speaker, the RightToLeft property has no value and should be left at its default. However, some languages such as Arabic and Hebrew run text from right to left. You would set RightToLeft for those languages so vertical scroll bars will appear on the *left* side of a TextBox.

The Scrollbars property enables you to add horizontal or vertical scroll bars to your TextBox so the user can employ them as a way of moving through text that exceeds the size of the TextBox. However, even without them, the user can always press the arrow keys, the PgUp and PgDn keys, the spacebar, and so on to move around through text that's not shown within the visible opening of the TextBox.

We looked at the TabIndex property in a previous session — it defines the order in which components get focus as the user repeatedly presses the Tab key to move among them.

The TabStop property, when set to False, removes the component from the TabIndex list. TabIndex is useful because it offers a quick way for the user to move among the input components (TextBoxes, CheckBoxes, and so forth) on a form — all without having to remove his or her hands from the keyboard and reach for the mouse to click a component into the focus. However, there are some components, like a PictureBox, that are not usually employed as user-input devices. So you can set their TabStop properties to False to eliminate them from the TabIndex group. Components such as Labels that can *never* be used as input devices simply have no TabIndex property in the first place, and are therefore never included in the tabbing.

How can a PictureBox ever be used as an input device, you ask? A simple example is that if the PictureBox is clicked at all, anywhere, something happens (because you put some programming into its Click event). You might display

```
        .Left = cmdSearch.Left
        .Top = txtTitle.Top
        .Width = cmdSearch.Width + 30
        .Height = txtTitle.Height + txtDesc.Height + 400
    End With

    With lstIndex
        .Left = cmdIndex.Left
        .Top = txtTitle.Top
        .Width = cmdIndex.Width + 30
        .Height = txtTitle.Height + txtDesc.Height + 400
    End With

    End Sub
```

Notice the fudge factors (+ 30 and + 400) in this code. Adjust those as necessary to make your ListBoxes neatly cover the buttons (except the Exit button).

The `ScaleHeight` property is similar to the `Height` property, but `ScaleHeight` doesn't include the form's Title bar or frame — just the interior measurement. We added 400 to take into account the space above and below the CommandButton.

The ListBoxes you put on your form cover up other components that you still want to work with. So, to make them invisible during design time, go to the Properties Window and change their `Left` properties to something like -8325. This moves them way off to the left of the screen — off into Virtual Land. (Just remember that you did this — so if you need them back for some reason, you can remove the minus sign.)

20 Min. To Go

Making Life Easier for the User

Users expect some shortcuts in their Windows applications, and woe be the programmer who forgets them. He or she will be thought unprofessional. Users expect an Alt+ shortcut key on each button, so they can press, for instance, Alt+S to activate the Search button without having to reach for the mouse.

They also expect to be able to press the Tab key to move among the components. The order of the `TabIndex` properties determines the order in which each component gets the focus. On this form, I would say that each button in turn going down should get the focus, followed by the Title TextBox and then the